eG Integration With Trouble Ticketing Systems

eG Enterprise v6

Restricted Rights Legend

Trademarks

Copyright

Table of Contents

Table of Figures

1

Introduction

eG Enterprise includes extensive monitoring capabilities for IT Infrastructure components. Problems detected by eG products can be reported to users in various ways – via the web, over email, and via SNMP traps to any SNMP console. Many enterprises use **Trouble Ticketing** (TT) systems to track problems with their IT infrastructures. Besides tracking the current problems, a trouble ticketing system enables an operator to dispatch service requests to the appropriate maintenance personnel. Maintenance personnel can use the trouble ticket system to update and monitor the status of current problems and follow these through to final resolution.

The integration of eG Enterprise with TT systems facilitates the following actions to be automatically performed in the TT system based on the open alarms in eG Enterprise:

- trouble tickets to be opened in the TT systems as and when a new alarm is detected by eG Enterprise;
- trouble tickets to be modified as and when an existing alarm is modified in eG Enterprise;
- trouble tickets to be closed as and when an alarm is removed in eG Enterprise;

1.1 How the eG Enterprise to TT System Integration Works

1.1.1 Alarms in eG Enterprise

To understand how the integration of the eG manager with a trouble ticketing system works, lets first consider what is an alarm. An alarm in eG Enterprise, is identified by an **Alarm ID**. At any given instant of time, an **Alarm ID** is a unique combination of the following attributes:

a. The problem component-type

b. The problem component (i.e., network device, application, etc.)

c. The problem layer

d. The problem priority (Critical, Major, Minor)

The eG monitoring interface lists alarms that currently exist in the eG Enterprise system. The goal of the eG Enterprise integration with TT systems is to be able to forward updated information on current alarms to the TT system.

Every time there is a state change (e.g., change of priority or correction of a problem) detected in the monitored environment, the eG manager checks the combination of component, component-type, layer, and priority combination for all open problems with their previous values to determine whether a new alarm has been generated, an existing alarm has been modified, or whether an existing alarm has been closed. If a new alarm has been generated, the eG manager assigns a distinct alarm ID for this alarm. If an existing alarm has been modified or closed, the eG manager retains the earlier assigned alarm ID for this alarm. Modification of an alarm can include any

of the following cases:

- A change in the alarm priority: This could be a switch to a higher or lower priority.
- A change in the alarm description: For example, originally, a usage-related alarm may have been raised on disk 'D' of a server. Later, disk 'C' of the same server might have experienced a space crunch, causing another alarm to be raised. In this case, the description of the original alarm will change to indicate that both disks C and D are experiencing a problem, but the alarm ID will not change. Changes in alarm description may also happen if additional tests being run for the same layer indicate a problem. A change may involve either an addition to the description (as in the example above) or a removal of one or more descriptors (e.g., the space usage of disk 'C' in the example above returning to a normal condition).
- A change in the list of impacted services

Each alarm is associated with a start date and time. The start date and time signifies when the alarm was first generated by the eG manager. Any change in the state of the alarm during a subsequent time does not cause a change in the start date and time of the alarm. Hence, even if an alarm changes in priority at a later time, its start date and time remain the same, until the alarm is finally closed. When an alarm is closed, a normal alert is generated, which will bear the current date and time.

In order to avoid conflicts/duplication of alarm IDs generated by each of the managers in a redundant eG manager cluster, the alarm ID is expressed as a string that is of the form ***<eG_Manager>_<numeric_value>***, where the ***<numeric_value>*** is a timestamp of when the alarm was first generated.

Prior to generating an alarm, the eG managers in a cluster synchronize with each other to ensure that duplicate alarms are not generated or that different alarm IDs are not generated for the same problem. As in the case of email alerts and SNMP traps, each manager in the cluster is responsible for generating alarms for agents that are directly reporting to the manager.

1.1.2 Integration with Trouble Ticketing Systems

The eG manager can be configured so that whenever an alarm undergoes a change – either generation, modification, or closure – the manager communicates this information to a TT system.

This communication can be in any of the forms mentioned below:

- In recent times, many trouble ticketing systems have been found to embed a unique mail interface that receives email alerts of problems in the environment. The eG Enterprise system can be configured to use this interface to send alarms generated by the eG manager as email alerts to the trouble ticketing system. Based on the mails so received, the trouble ticketing system may generate trouble tickets and forward them to the concerned maintenance personnel.
- The eG Manager supports a command line interface, that can be configured to automatically execute TT system-specific commands as and when alarms are added, modified, or deleted in eG Enterprise. This interface offers a way of communication between the eG Manager and a TT system.
- The eG Manager can also forward its alarm information to any web services interface that the Trouble Ticketing System may support to trigger the automatic creation/closure (as the case may be) of trouble tickets.

Each chapter in this document discusses each of these integration modes in detail.

1.1.3 Handling eG Alarms in a Trouble Ticketing System

A trouble ticket system must be configured to process alarms reported to it by an eG manager. The alarm ID must be used to uniquely identify an alarm. The functions that the TT system must perform are:

- Determine if an alarm ID indicates a new alarm. If yes, open a new trouble ticket.
- If an alarm ID indicates an existing alarm, check the priority of the alarm. If the priority is **Normal**, this implies that the alarm has been closed in eG Enterprise. Hence, close the corresponding trouble ticket in the TT system.
- If an alarm ID indicates an existing alarm and the priority of the alarm is not **Normal**, update the corresponding trouble ticket with the current priority of the alarm and with its current description.

These functions often involve scripting/configurations on the TT system.

Once the above steps are accomplished, by reviewing the status of the trouble tickets, administrators can be immediately aware of the current status of the infrastructure being monitored, without having to login to the eG Enterprise console.

- If a standalone (i.e., non-redundant) eG manager is restarted, all outstanding alarms and hence, all open trouble tickets will be closed. After the restart, if an old problem re-occurs, the restarted manager will assign a new alarm ID to this problem; as a result, new trouble tickets will be opened for such problems.

- In a redundant configuration, when a manager is restarted, it checks if the other manager is available. If the other manager in the cluster is not available, all outstanding alarms will be closed. On the other hand, if the other manager in the cluster is available, then the manager being restarted will synchronize alarm information with the other manager. When it detects a problem, the restarted manager checks to see if the other manager in the cluster has already assigned an alarm ID to this problem. If so, then the restarted manager assigns the same ID to the problem. In such a case, new trouble tickets will not be opened for the existing problems.

- In rare instances, when there are rapid alarm transitions (eg., from critical to normal to critical state) for the same component type-component name-layer-priority combination in a redundant eG manager configuration, the same alarm ID may be re-used to refer to the new alarm.

- The same eG manager can be configured to different modes of integration with a TT system – be it email integration, command line integration, or web services-based integration.

- Since the eG manager forwards the current status of an alarm to the TT system, and since such transmission is done only at periodic intervals, the eG Enterprise-TT system integration does not capture all state transitions in the infrastructure being monitored. For instance, if the **MailCheckPeriod** setting is 3 mins, an event that happens and gets corrected within 1 min is never captured in the TT system. Consider changing the **MailCheckPeriod** setting to a lower value (upto 1sec), if you require higher sensitivity in trouble ticket tracking. Obviously, lower the value of the **MailCheckPeriod**, greater is the overhead on the eG manager.

2

Trouble Ticket Integration Using the TT Mail Interface

The eG manager can be configured so that whenever an alarm undergoes a change – either generation, modification, or closure – the manager communicates this information to a TT system. This communication can be in the form of formatted email messages that can be processed by a TT system using email interfaces that it supports.

This communication will take effect **only if the eG license enables trouble ticket integration**.

Before performing the configuration, ensure that a mail server and an admin mail id have been configured for the eG Enterprise system. Refer to the *eG User Manual* to know how to configure the mail server.

To direct email alerts generated by the eG manager to the TT system, open the **eg_services.ini** file (in the **<EG_INSTALL_DIR>\manager\config** directory). Specify the following in the **[TTMAIL]** section of the file.

```
[TTMAIL]

MailSubject=

MailTo=

OutputFormat=<eG_Alarm>\n<Priority>$prior</Priority>\n<AlarmId>$alarmid</AlarmId>\n<User>$user</User>\n<ComponentName>$cname</ComponentName>\n<ComponentType>$ctype</ComponentType>\n<Layer>$layer</Layer>\n<Problem>$pdesc</Problem>\n</eG_Alarm>

SeparateMails=No

IncludeAllCompUser=No

AllowedAlarms=critical,major,minor

UseUniqueId=Yes

PriorityAsNumeric=No

Critical=1

Major=2

Minor=3

Normal=0

ApplyPriorityForAllTTMails=No

TestsList=All
```

```
SendNormalTTMails=Yes
```

Against the **MailSubject** parameter, provide an appropriate subject for the alert mail. To ensure that the mail subject reflects the problem component name, problem component type, the problem priority, etc., variables can be used in the **MailSubject** definition, in the following format: *$alarmid#$user#$cname#$ctype#$layer#$prior#$pdesc*.

The variable *$alarmid* in the **MailSubject** ensures that the subject of the TT mail contains the alarm id. Similarly, the *$user, $cname, $ctype* variables represent the user with whom the alarm is associated, the name of the problem component, and the problem component type, respectively. Likewise, the *$layer, $prior, and $pdesc* variables denote the problem layer, the alarm priority, and the alarm description, respectively. The '#' is the separator, but it can be changed. You can rearrange the order of the variables, and even omit a few variables if need be; but, the variable names and the '$' symbol preceding the name should not be changed. A sample subject has been provided below:

```
1#john#printer:NULL#Host_system#NETWORK#Critical#Network connection down
```

This mail subject has been described below:

- **1** is the **alarm id**
- **john** is the user with whom the problem component is associated
- **printer:NULL** is the hostname of the problem component
- **Host_system** is the component type
- **NETWORK** is the layer name
- **Critical** is the alarm priority
- **Network connection down** is the alarm description

Also, note that such a mail subject specification will work only if the **SeparateMails** flag in the **[TTMAIL]** section is switched on. By default, this flag is set to **No**, indicating that a single TT mail will comprise of details pertaining to all the alarms that were raised by the eG Enterprise system during that point in time. In such a case, the variables in the **MailSubject** cannot be substituted by the corresponding problem information; in this case therefore, by default, **eGTTMail** will appear as the subject of the TT mail. However, if every problem event in the environment should generate a separate TT mail, then set the **SeparateMails** flag to **Yes**. In such a case, the variables in the **MailSubject** will be substituted by the corresponding details from the problem information.

Next, specify the mail id to which the alerts need to be delivered against the **MailTo** parameter. Multiple mail IDs can also be specified as a comma-separated list.

The **OutputFormat** parameter signifies the format in which the alarm content needs to be mailed. The default contents of a TT mail are characterized by the following **OutputFormat** specification:

```
<eG_Alarm>\n<Priority>$prior</Priority>\n<AlarmId>$alarmid</AlarmId>\n<User>$user</User>\n<ComponentName>$cname</ComponentName>\n<ComponentType>$ctype</ComponentType>\n<Layer>$layer</Layer>\n<Problem>$pdesc</Problem>\n</eG_Alarm>
```

As stated earlier, a single TT mail can comprise of details pertaining to numerous alarms. Every such alarm definition within a TT mail will typically begin with the tag *<eG_Alarm>* and end with the tag *</eG_Alarm>*. This essentially indicates that the details contained within these tags pertain to a single alarm. These tags can be changed if so required. For example, you can specify *<Alarm_info>* and *</Alarm_Info>* instead of *<eG_Alarm>* and *</eG_Alarm>*. The variables **$prior**, **$alarmid**, **$user**, **$cname**, **$ctype**, **$layer**, **$pdesc**, during run-time, will display the alarm priority, alarm id, the user associated with the problem component, the problem component's name, the problem component type, the problem layer, and the problem description, as the case may be. In a TT mail, the value of each of the defined variables will be enclosed within the opening and closing tags defined in the **OutputFormat**. For example, take the case of the specification *<Priority>$prior</Priority>*. In the TT mail for a *critical* alarm, this specification will appear as *<Priority> critical </Priority>*. These tags serve as qualifiers for the

enclosed values. In other words, they indicate what value is displayed within. These tags can also be modified, if need be. However, the dollared variable names cannot be changed. '**/n**' acts as a separator for the values. A sample TT mail output has been provided below:

```
<eG_Alarm>
<Priority>Normal</Priority>
<AlarmId>2</AlarmId>
<User>john</User>
<ComponentName>king:7001</ComponentName>
<ComponentType>WebLogic_server</ComponentType>
<Layer>WL_SERVICE</Layer>
<Problem>Many invocations{DD}</Problem>
</eG_Alarm>
```

If the problem component is associated with multiple users, then the **$user** variable will display a comma-separated user list. Typically, a component will be associated with those users who are responsible for resolving the issues with it. However, some users will be authorized by the eG Enterprise system to oversee the performance of all the components in the environment (for eg., the default users of the eG Enterprise system – *admin* and *supermonitor*). In most cases, the responsibilities of such users will be more 'supervisory' in nature, and not administrative – i.e. might not involve troubleshooting and problem redressal. Therefore, by default, the eG Enterprise system will hide the ID of this user from the user list displayed in the TT mail output. To ensure that the user list displays such a user ID too, set the **IncludeAllCompUser** parameter in the **eg_services.ini** file (in the **<EG_INSTALL_DIR>\manager\config**) to **Yes**. By default, this parameter will be set to **No**.

So, when an email alert is received by the TT system, the TT system generates a corresponding trouble ticket based on the information sent via mail. The email interface of the TT system will record and maintain the alarm id and trouble ticket id mapping, so that when the status of an existing alarm changes (for example – change in alarm priority, change in alarm description, etc.) the corresponding trouble ticket id is automatically updated with the change. Similarly, if an alarm is closed, then upon receipt of an email reflecting the closure of the alarm, the corresponding trouble ticket id will also be automatically removed by the TT system. However, if you provide a comma-separated list of alarm priorities against the **AllowedAlarms** parameter, you can instruct the eG manager to feed the TT system with only those email alerts that pertain to the specified priorities. This additionally ensures that Normal mail alerts are sent by the eG Enterprise system only when the alarms of the specified priorities are closed. For example, say that the **AllowedAlarms** parameter contains the value **critical**. This indicates that the TT system will be alerted of only the critical issues in the eG Enterprise system. This also indicates that the TT system will receive Normal mail alerts from the eG manager only when the **critical** alarms are closed, and not the **major** or **minor** alarms.

Also, by setting the **ApplyPriorityForAllTTMails** flag to **Yes** or **No**, you can indicate whether the **AllowedAlarms** setting applies only to each new alarm ID that is raised by the eG manager or to modified alarms as well. As already stated, if an existing alarm has been modified, the eG manager retains the earlier assigned alarm ID for this alarm. The following are considered alarm modifications:

- A change in the alarm priority: This could be a switch to a higher or lower priority.
- A change in the alarm description: For example, originally, a usage-related alarm may have been raised on disk 'D' of a server. Later, disk 'C' of the same server might have experienced a space crunch, causing another alarm to be raised. In this case, the description of the original alarm will change to indicate that both disks C and D are experiencing a problem, but the alarm ID will not change. Changes in alarm description may also happen if additional tests being run for the same layer indicate a problem. A change may involve either an addition to the description (as in the example above) or a removal of one or more descriptors (e.g., the space usage of disk 'C' in the example above returning to a normal condition).
- A change in the list of impacted services

If the **ApplyPriorityForAllTTMails** flag is set to **Yes**, then the eG manager will send an email alert into the TT system

even if one of the above modifications occur on an existing alarm, as long as the priority of the modified alarm belongs to the list of priorities configured against **AllowedAlarms**. On the other hand, if the **ApplyPriorityForAllTTMails** flag is set to **No**, then the eG manager will send email alerts only for every new alarm ID - i.e., for the first alarm of an ID. In this case, the manager will ignore all subsequent changes to the priority of the alarm.

In the case of a redundant setup, the TT mail output will slightly differ. Given below is the sample output of a TT mail in a redundant manager setup.

```
<eG_Alarm>
<Priority>Minor</Priority>
<AlarmId>192.168.10.7_1125999629278</AlarmId>
<User>balaji</User>
<ComponentName>apache7:7077</ComponentName>
<ComponentType>Web_server</ComponentType>
<Layer>WEB_TRANSACTIONS</Layer>
<Problem>High request rate{Current_alarms_page/gopi}</Problem>
</eG_Alarm>
```

Note the difference in the format of the **AlarmId**. This variable, which is typically a numeric value, has been expressed as *<ManagerIP>_<a long value>*. This is done to prevent duplication of alarm IDs across the multiple managers in a redundant manager cluster.

By default, alarm IDs generated by the eG manager start from 1 whenever the manager restarts. Therefore the **UseUniqueId** parameter in the **[TTMAIL]** section of the **eg_services.ini** file (in the **{EG_INSTALL_DIR}\manager\config** directory) is set to **Yes**, by default. In order to ensure that alarm IDs are not re-used (i.e., do not start from 1) when the manager restarts, set this parameter to **No**.

If need be, you can make sure that the TT mails indicate the alarm priority using numbers instead of priority names such as *critical, major, minor,* or *normal*. For this purpose, you will have to set the **PriorityAsNumeric** flag in the **[TTMAIL]** section to **Yes**. By default, this flag is set to **No**, indicating that the priority of an alarm is indicated using the priority name by default. If this flag is set to **Yes** instead, then the default priority name-number mappings defined in the **[TMAIL]** section will automatically apply. These default mappings are as follows:

```
Critical=1

Major=2

Minor=3
```

According to these mappings, if the **PriorityAsNumeric** flag is set to **Yes**, then, in every TT mail sent subsequently, critical priority will be represented by number 1, major priority by number 2, and minor priority by number 3. If required, you can even change the numbers that should represent the alarm priorities. For instance, your priority name-number mapping can be as follows:

```
Critical=10

Major=11

Minor=12
```

You can even configure the specific tests for which TT mails are to be sent using the **TestsList** parameter in the **[TTMAIL]** section. By default, this parameter is set to *All*, indicating that the eG manager, by default, sends out TT mails for alarms related to all tests. To restrict TT mail transmission to specific tests, provide a comma-separated list of tests against **TestsList**. While providing test names here, make sure you provide the *<internaltestnames>* and not

the display names. For instance, say, you want TT mails to be sent only when the eG manager raises alarms for the **Processes** test and the **SystemDetails** test. To achieve this, your **TestsList** specification should be as follows:

```
TestsList=ProcessTest,SystemTest
```

In the specification above, the internal name for **Processes** test is *ProcessTest*, and the same for **SystemDetals** test is *SystemTest*. To determine the *internal name* of a test, do the following:

1. Open the **eg_lang*.ini** file (from the **<EG_INSTALL_DIR>\manager\config** directory), where * is the language code that represents the language preference that you have set using the **USER PROFILE** page. In this file, the component types, measure names, test names, layer names, measure descriptions, and a wide range of other display information are expressed in a particular language, and are mapped to their eG equivalents.
2. Now, search the **eg_lang*.ini** file for the test name of interest to you. For example, to know the internal name of the **SystemDetails** test, search the language file for the text, **SystemDetails**.
3. Since the eG equivalent of the **SystemDetails** test is *SystemTest*, you will find a specification to that effect in the **[TEST_NAME_MAPPING]** section of that file. For our example, the specification would be as follows:

   ```
   SystemTest=SystemDetails
   ```

In addition to the above, a **SendNormalTTMails** flag also exists, which is set to **Yes** by default. This indicates that, by default, the eG manager sends out TT mails when a problem is resolved. To turn off this capability, set the **SendNormalTTMails** flag to **No**.

3

Trouble Ticket Integration Using the eG TT CLI

The eG manager can also be configured so that whenever it detects a new alarm, a change in an existing alarm, or a closure of an existing alarm, it executes a command with the appropriate parameters indicating the current status of the alarm. **Note that this capability is available for stand-alone Windows managers, and Windows managers operating in redundant clusters only.** Prior to configuring this capability on such managers, the following pre-requisites need to be fulfilled:

- The eG trouble ticket manager capability should be enabled in the eG Enterprise license.
- In case of a redundant manager configuration, all the eG managers that are part of the cluster should have the trouble ticketing integration capability enabled in their respective licenses.
- To receive detailed diagnosis information via CLI, the eG license should enable the **Detailed Diagnosis** capability; in case of a redundant manager configuration again, all the eG managers that are part of the cluster should have the detailed diagnosis capability enabled in their respective licenses.

To configure the command to be executed, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. From the **MANAGER SETTINGS** panel to the left of the page that appears, select the **TT Manager** option.
4. The contents of the right panel will then change to display the **Trouble Ticket (TT) Manager CLI** section as depicted by Figure 3.1 below. .

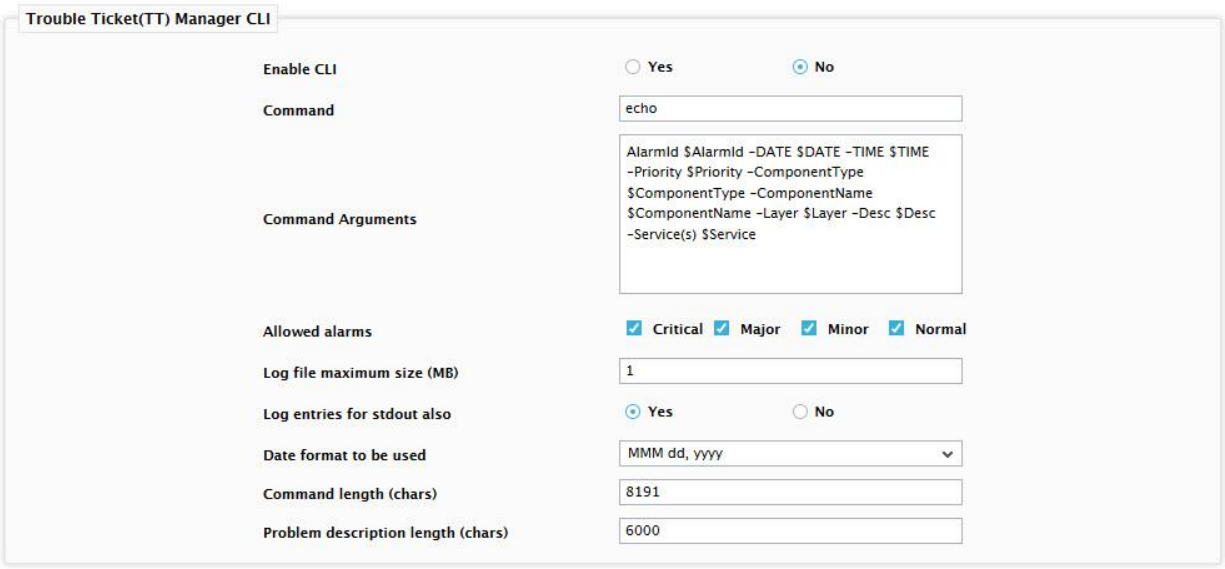

Figure 3.1: eG Manager settings for TT System CLI integration

5. Set the **Enable CLI** parameter in Figure 3.1 to **Yes** to enable this capability.
6. In the **Command** text box, **echo** is displayed by default, indicating that the eG manager will execute an *echo* command by default to communicate with the TT system.
7. The **Command Arguments** text box displays the default input parameters that the **echo** command takes during execution. These default parameters are as follows:

AlarmId $AlarmId -DATE $DATE -TIME $TIME -Priority $Priority -ComponentType $ComponentType -ComponentName $ComponentName -Layer $Layer -Desc $Desc –Service(s) $Service

As you can see, the each parameter is represented by a qualifier and a variable name. While the qualifier is typically prefixed by a hyphen (-), the variable name is prefixed by a $ symbol. These variables will be substituted by actual values during runtime. Using the qualifiers, you will be able to tell what value follows. For instance, at runtime, the parameter ***–Priority $Priority*** could appear as ***–Priority Critical.*** This implies that the **Priority** of the problem is **Critical**.

- You can alter the qualifier if you need to, but the variable names (the $ preceded strings in the previous example) should not be changed.
- The position of the qualifier-variable pairs can be changed in the command line – for instance, you can move the ***–Service $Service*** parameter to appear next to the ***AlarmID $AlarmId*** if you want to.
- Though the eG manager executes a command line by default, you can change this **Command** specification to execute a batch file/script file/executable instead.
- The **Command Arguments** specification can include any special character that will work from the Windows command prompt or Unix shell, except the # (hash) character, which is used as a separator between the command and the arguments.
- On Windows, the eG manager runs as a service, and hence, has access to all the sysem-defined environment variables. You can use these variables as required in the script that is invoked by the **Command** specification.

Given below is the list of parameters the default **Command Arguments** display takes, and a brief description of each parameter:

- ***AlarmId $AlarmId*** – unique identifier of the alarm
- ***-DATE $DATE*** – the date on which the problem occurred
- ***-TIME $TIME –*** the time at which the problem occurred.
- ***-Priority $Priority*** – the problem priority - whether Critical, Major, Minor, Normal
- ***-ComponentType $ComponentType*** – The problem component type
- ***-ComponentName $ComponentName*** – The problem component name
- ***-Layer $Layer*** – the protocol layer to which the problem relates.
- ***-Desc $Desc*** – a brief description of the problem. This will include a pipe (|) separated list of the following fields – the site name (if relevant to the test), test name, the alarm string (a textual description of the problem), and measurement host. The description is not applicable if the alarm severity is **Normal**.
- ***–Service(s) $Service*** – If more than one service is impacted, this will include a comma-separated list of services.
- **-DD $DD** - **This parameter is not available by default**. If required, you can configure this parameter additionally for the command so that, the output includes detailed diagnosis information. In the output, ***$DD*** will be represented in the following format:

"DDcolumn1 DDcolumn2 DDcolumn3 …~#~DDdata1~!~DDdata2~!~DDdata3~!~..."

In the eG user interface, the detailed diagnosis pertaining to a single test/measure/descriptor combination is typically presented in a tabular format, with rows and columns. Accordingly, in the command output for a specific test-measure-descriptor combination, the variables *DDcolum1, DDcolumn2*, etc., will be substituted by the names of the columns in the detailed diagnosis, and the variables *DDdata, DDdata2, etc.,* will report the values that correspond to each column.

- The detailed diagnosis information reported in the CLI output will typically be the DD information available in the eG database, at the time of command execution by the eG manager.
- A single alarm could have multiple tests, measures, and descriptors associated with it. For each test-measure-descriptor combination, there will be a corresponding DD entry in the command output line (subject to the length restriction discussed in the **Limitations** section below). The DD output for a single test-measure-descriptor combination will include the column names and data. In the output, these columns and their corresponding data will be separated using the separator ~#~. If a specific combination does not have DD configured or there is no DD reported for that descriptor, a value "-" will be reported.
- The DD output for each test-measure-descriptor combination will be separated using #~#.
- Detailed diagnosis information for a test-measure-descriptor combination could include rows of data. In the command output, the separator !~! is used to separate multiple rows of DD data.
- Each row of data in the DD would report values for several columns. The values that correspond to each column will be separated using ~!~ in the DD output.

A sample standard output of the default **Command** specification is given below:

```
AlarmId "192.168.10.133_1264839507765" -DATE "Jan 30, 2010" -TIME "13:48:24" -
Priority "1" -ComponentType "Host system" -ComponentName "win: NULL" -Layer
"Operating System" -Desc "-|SystemDetails|High CPU
utilization{Processor_0}|win,-|SystemDetails|Free memory is
low{Processor_0}|win" -Service "-" -DD " PID %CPU ARGS
~#~692~!~1.60~!~csrss!~!760~!~0.53~!~services!~!6960.53~!~ js #~# PID %MEM
ARGS~#~4032~!~7.83~!~tomcat!~!2112~!~6.33~!~firefox!~!3472~!~5.52~!~dbvis"
```

According to the above output, the following alarm information will be sent to the third-party TT system:

Qualifier	Variable
AlarmId	192.168.10.133_1256878963888
DATE	30/10/2009
TIME	10:32:43
Priority	Critical
ComponentType	Generic
ComponentName	gen133:NULL
Layer	Application Processes
Desc	In our example, the command line output clubs the information pertaining to two alarms related to a single test. The description of the first alarm indicates the following: ▪ the site affected (if applicable) : In the case of our example, no web site has been impacted by the Critical problem; therefore, only a '-' (hyphen) is displayed instead in the output line. ▪ The test that reported the problem: In the case of our example, this is **SystemDetails** test ▪ The alarm description: In the case of the sample output, this is - **High CPU utilization { Processor_0}** ▪ The measurement host: In the case of the sample output, this is – **win** The description of the second alarm includes the following: ▪ the site affected (if applicable) : In the case of the second alarm also, no web site has been impacted by the Critical problem; therefore, only a '-' (hyphen) is displayed instead in the output line. ▪ The test that reported the problem: In the case of our example, this is **SystemDetails** test ▪ The alarm description: In the case of the sample output, this is - **Free memory is low{Processor_0}** ▪ The measurement host: In the case of the sample output, this is

	– **win**
Service	Since no service has been impacted by the problem at hand, only a '-' is displayed here instead
DD	The sample output above includes the DD information pertaining to two test-measure-descriptor combinations. For the first test-measure-descriptor combination - i.e., for the **CPU utilization** measure of the descriptor **Processor_0** reported by the **SystemDetails** test - the DD output includes the following: ▪ The columns in the DD are as follows: **PID**, **%CPU**, and **ARGS** (as in 'arguments'). ▪ The DD for this test-measure-combination includes three rows of data. In the first row of data, the following values will be reported: ▪ **692** as the **PID** ▪ **1.60** as the **CPU%** ▪ **csrss** as the **ARGS** ▪ In the second row of data, the following values will be reported: ▪ **760** as the **PID** ▪ **0.53** as the **CPU%** ▪ **services** as the **ARGS** ▪ In the third row of data, the following values will be reported: ▪ **696** as the **PID** ▪ **0.53** as the **CPU%** ▪ **js** as the **ARGS** For the second test-measure-descriptor combination - i.e., for the **Free memory** measure of the descriptor **Processor_0** reported by the **SystemDetails** test - the DD output includes the following: ▪ The columns in the DD are as follows: PID, %MEM, and ARGS (as in 'arguments'). ▪ The DD for this test-measure-combination includes three rows of data. In the first row of data, the following values will be reported: ▪ **4032** as the **PID** ▪ **7.83** as the **MEM%** ▪ **tomcat** as the **ARGS** ▪ In the second row of data, the following values will be reported: ▪ **2112** as the **PID** ▪ **6.33** as the **MEM%** ▪ **firefox** as the **ARGS** ▪ In the third row of data, the following values will be reported:

	▪ **3472** as the **PID** ▪ **5.52** as the **MEM%** ▪ **dbvis** as the **ARGS**

- DD information will be formatted for better display in eG user interface. From the CLI however, no such special formatting can be effected on the DD output.
- If an alert is raised on multiple descriptors, hyphen (-) will be provided if DD is not available for a specific descriptor.
- Detailed diagnosis could have special characters (e.g., double quotes) that may require special handling when passed to a CLI.
- If the command line execution fails when including DD, CLI will resume execution by excluding the DD, so that the alert does not get lost.

8. By selecting the required check boxes against **Allowed alarms**, you can indicate the alarm priorities for which the eG manager needs to execute the specified command.
9. To track the status of the command execution and to troubleshoot issues with the same, use the **ttexec.log** file that is automatically created in the **<EG_INSTALL_DIR>\manager\logs** directory. You can specify the maximum size up to which this log file can grow, in the **Log file maximum size** text box. When the file reaches the specified size limit, the details originally logged in the **ttexec.log** file will be moved to another log file named **ttexec.log.1**, and the newer information will be logged in the **ttexec.log** file instead. This log rotation mechanism helps ensure that the log file does not grow beyond control.
10. You can even indicate the type of information you want logged in the **ttexec.log** file. By default, the log files capture both the errors and the standard output of the specified **Command**; accordingly, the **Log entries for stdout also** flag is set to **Yes** by default. A sample log is provided below, where both standard output and errors have been logged:

```
03/11/2009 18:25:03 INFO AlarmId "192.168.10.133_1257252903826" -DATE "Nov 03,
2009" -TIME "18:25:02" -Priority "Critical" -ComponentType "Generic" -
ComponentName "gen133:NULL" -Layer "Application Processes" -Desc "-
|Processes|Process not running{cmd}|gen133" -Service "-"#AlarmId
"192.168.10.133_1257252903842" -DATE "Nov 03, 2009" -TIME "18:25:02" -Priority
"Major" -ComponentType "Generic" -ComponentName "gen133:NULL" -Layer "Application
Processes" -Desc "-|Processes|Process not running{notepad}|gen133" -Service "-"
03/11/2009 18:36:12 ERROR Executing command - ech AlarmId
"192.168.10.133_1257253386239" -DATE "Nov 03, 2009" -TIME "18:32:37" -Priority
"Normal" -ComponentType "Generic" -ComponentName "gen133:NULL" -Layer "Application
Processes" -Desc "-|Processes|Process not running{cmd}|gen133" -Service "-
"#AlarmId "192.168.10.133_1257253386255" -DATE "Nov 03, 2009" -TIME "18:32:38" -
Priority "Normal" -ComponentType "Generic" -ComponentName "gen133:NULL" -Layer
"Application Processes" -Desc "-|Processes|Process not running{notepad}|gen133" -
Service "-"
```

```
03/11/2009 18:36:12 ERROR 'ech' is not recognized as an internal or external
command,
```

If you want to capture only the errors, set the **Log entries for stdout also** flag to **No**.

11. From the **Date format to be used** list box, select the format in which the date/time of the problem should be reported in the command output.
12. Specify the maximum permissible length of the command in the **Command length** text box. By default, the command line can have a maximum of 8191 characters. You can alter this default setting by specifying a length of your choice in the **Command length** text box. If the actual command length exceeds the specified limit, then the output will not return the list of affected services and the detailed diagnosis information; instead, an empty string will appear next to the ***–Services*** qualifier. If the command length continues to exceed the specified limit even after truncating the services list and the DD, the command execution will return an error.
13. Specify the length of the problem description in the **Problem description length** text box. If the actual problem description exceeds the specified length, the characters that fall beyond the specified limit will be truncated.
14. Finally, click on the **Update** button to save the changes.

As described above, the eG manager offers a high degree of flexibility in the configuration of the command that should be executed. The periodicity at which the eG manager checks alarms and determines what information it should send to a TT system is determined by the setting of the **MailCheckPeriod** attribute in the **[MISC_ARGS]** section of the **eg_services.ini** file (in the **<EG_INSTALL_DIR>\manager\config** directory). By default, this value is 180 seconds (3 minutes).

4

Trouble Ticket Integration Using a Web Services Framework

eG Enterprise can integrate with TT systems that support a web services API. The eG manager establishes an HTTP/S connection to a web services URL on the third-party system and communicates alarm information to that TT system using its web services API. Upon receipt of an alarm, the TT system automatically generates/modifies/closes trouble tickets.

Without the need for any complex instrumentation, eG Enterprise can readily integrate with Manage Engine's ServiceDesk through its web services interface. Section 4.1 of this chapter describes the procedure to integrate with ServiceDesk.

To integrate with any other help desk system that supports a web services interface, you can easily fine-tune eG's web services framework to suit your needs. Section 4.2 of this chapter describes how this can be achieved.

4.1 Integrating with ManageEngine's ServiceDesk

ManageEngine ServiceDesk is a comprehensive Help Desk and Asset Management software that provides help desk agents and IT managers an integrated console to monitor and maintain the assets and IT requests generated from the users of the IT resources in an organization.

To integrate the eG manager with ServiceDesk, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. From the **MANAGER SETTINGS** panel to the left of the page that appears, select the **TT Manager** option.
4. Scroll down the right panel to view the **Trouble Ticket (TT) Integration** section (see Figure 4.1).

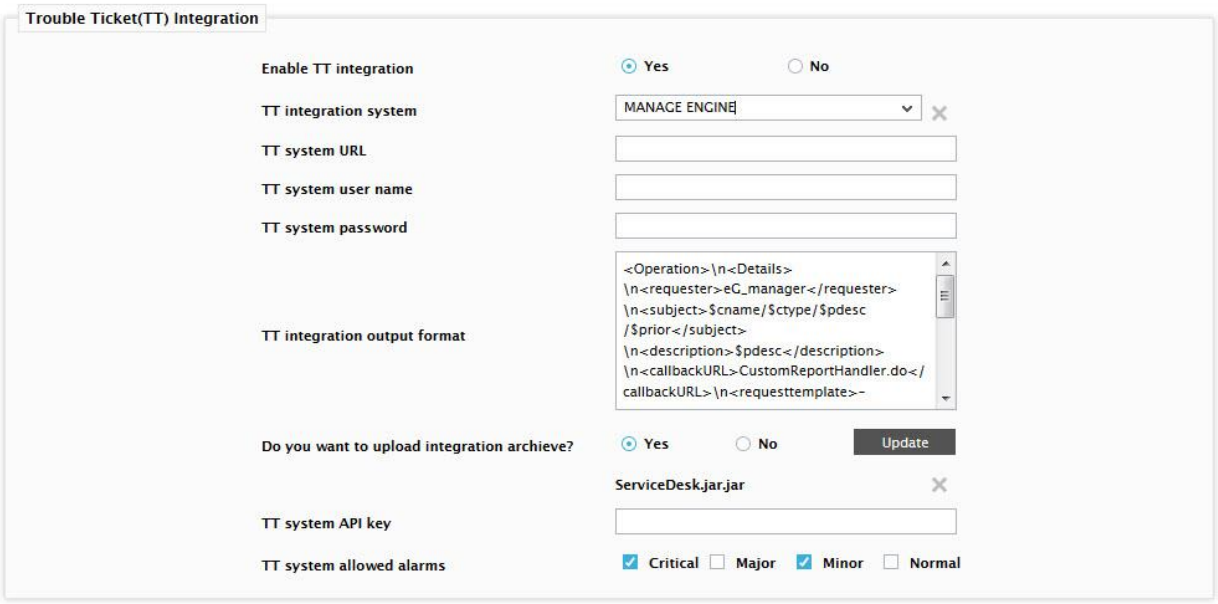

Figure 4.1: Integrating the eG manager with ManageEngine's ServiceDesk

5. To enable integration via the web services interface of ServiceDesk, set the **Enable TT integration** flag in Figure 4.1 to **Yes**. By default, this is set to **No**.

6. From the **TT integration system** drop-down, select **MANAGE ENGINE** as the TT system with which the eG manager should integrate.

7. Then, against the **TT system URL**, specify the Web Services Description Language (WSDL) URL via which the eG manager should connect to ServiceDesk's web services interface. This URL should be of the following format: *http://<ServiceDeskservername>:<port number>/sdpapi/<module>*

8. If the connection needs to be authenticated, then provide a valid user name and password against **TT system user name** and **TT system password** text boxes, respectively.

9. ServiceDesk's web services API, known as REST API, is capable of interpreting problem inputs it receives and automatically generating/updating trouble tickets, only if the inputs are in XML format. This is why, by default, the eG manager sends its alarm output in XML format to ServiceDesk. The standard format, as displayed against **TT integration output format** in Figure 4.1, is as follows:

```
<Operation>\n<Details>\n<requester>eG_manager</requester>\n<subject>$cname/$ctype/
$pdesc/$prior</subject>\n<description>$pdesc</description>\n<callbackURL>CustomRep
ortHandler.do</callbackURL>\n<requesttemplate>-
</requesttemplate>\n<priority>$prior</priority>\n<layer>$layer</layer>\n<group>$us
er</group>\n<technician>eG_manager</technician>\n<level>1</level>\n<status>$status
</status>\n<service>$Service</service>\n</Details>\n</Operation>
```

The text enclosed within angular brackets – eg., *<Operation>* - are the XML tags that ServiceDesk's web services API recognizes. **These tags cannot be changed**. The text enclosed within an opening and a closing tag can either be static text or a variable. These tags and the values they contain are discussed hereunder:

<Operation>|n<Details>|n	The operations performed with REST API are based on the '**operation**' parameter and is sent to the url via HTTP POST method.
<requester></requester>	Should contain a static text indicating the user requesting for a

	trouble ticket. Using ServiceDesk's self-service portal, you can add, edit, or remove requesters. In the case of the standard output format above, a requester named *eG_manager* has apparently being created. Therefore, the static text *eG_manager* is enclosed within these tags. If the name of the requester is changed in ServiceDesk, make sure that this text is also changed.
<subject></subject>	Should contain a slash-separated list of variables representing the subject of the trouble ticket. In the case of the standard output format above, the following variables have been used in the subject and this is what they denote: ▪ $cname – at run time, this variable will change to display the exact name of the problem component ▪ $cname – at run time, this variable will change to report the problem component-type ▪ $pdesc- at run time, this variable will change to report the problem description. ▪ $prior – at run time, this variable will change to report the problem priority You can add / remove the subject variables at will, but you cannot **change the variable representation**. In other words, **you cannot change $cname to #name** when defining the output format.
<description></description>	Should contain the variable that represents the problem description – i.e., **$pdesc**. At run time, this variable will change to report the precise problem description.
<callbackURL</callbackURL>	Provide a valid callback URL within these tags. When the cause for an eG alarm is resolved, ServiceDesk will invoke this URL. The URL functions as a notification to the eG manager indicating that the ticket is resolved. If this URL (callback URL) is not provided, ServiceDesk will not perform any additional operation on the ticket.
<requesttemplate></requesttemplate>	Within these tags, specify the name of the request template to be used (if any). If no request template applies, as in the case of our example above, then enclose a – (hyphen) within these tags. Using ServiceDesk, one can create different incident/request templates, each configured with a set of fields using which an incident is to be reported.
<priority></priority>	Should contain the variable that represents the problem priority – i.e., **$prior**. At run time, this variable will change to report the precise problem priority.
<layer</layer>	Should contain the variable that represents the problem layer – i.e., **$layer**. At run time, this variable will change to report the problematic layer.
<group></group>	Should contain the variable that represents the user/support group to which the technician responsible for resolving this alarm belongs.

	At run time, this variable will change to report the exact user group.
<technician></technician>	Should contain the static text that indicates the name of the technician responsible for resolving the alarm. In the case of our standard output format, *eG_manager* is the technician. You can change this to reflect the name of any other technician who has been configured in ServiceDesk. .
<level></level>	Should indicate the support level. In the case of our standard output format, the support level is *1*.
<status></status>	Should contain the variable **$status** that represents the status of the ticket. At run time, this variable will change to report the exact status.
<service></service>	Should contain the variable **$service** that represents the name of the business service impacted by the problem. At run time, this variable will change to to report the correct business service name.

10. The implementation of eG integration with ServiceDesk exists in a Java class file that is jarred with the REST API files required for connecting to ServiceDesk and a set of library files that support the implementation class file. In eG, this jar file is referred to as the **TT integration archive file**. This jar file, named **ServiceDesk.jar**, is available in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. If changes are made to one/more class/API/library files that are jarred to **ServiceDesk.jar**, you will have to regenerate the jar file and re-upload it to the eG manager. In such a case, set the **Do you want to upload the integration archive?** flag to **Yes**, and click the **Update** button alongside. This will invoke Figure 4.2.

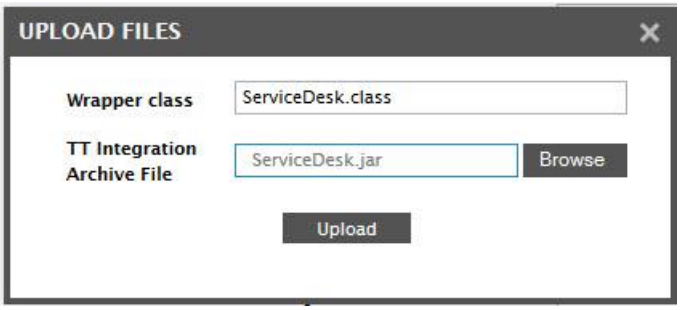

Figure 4.2: Uploading the TT integration jar file to the eG manager

Here, specify the name of the **Wrapper class** of the TT integration archive file, and provide the full path to the **TT Integration Archive File** to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button.

11. Then, specify the **TT system API key**. The authentication between ServiceDesk and the eG manager through this API key. ManageEngine generates a unique key for the user, whose credentials are specified against **TT system user name** and **TT system password** parameters in Figure 4.1 above. The eG manager will not be able to use the ServiceDesk API if this key is not specified or an invalid key is specified.
12. Then, indicate what type of eG alarms need to be forwarded to ServiceDesk by selecting the check boxes corresponding to the alarm priorities listed in the **TT system allowed alarms.** Trouble tickets will be generated by ServiceDesk only for the alarms of the chosen priorities.
13. Finally, click the **Update** button.

4.2 Integrating with any Third-party TT System that Supports a Web Services Interface

To achieve this, follow the steps below:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. From the **MANAGER SETTINGS** panel to the left of the page that appears, select the **TT Manager** option.
4. Scroll down the right panel to view the **Trouble Ticket (TT) Integration** section (see Figure 4.1).

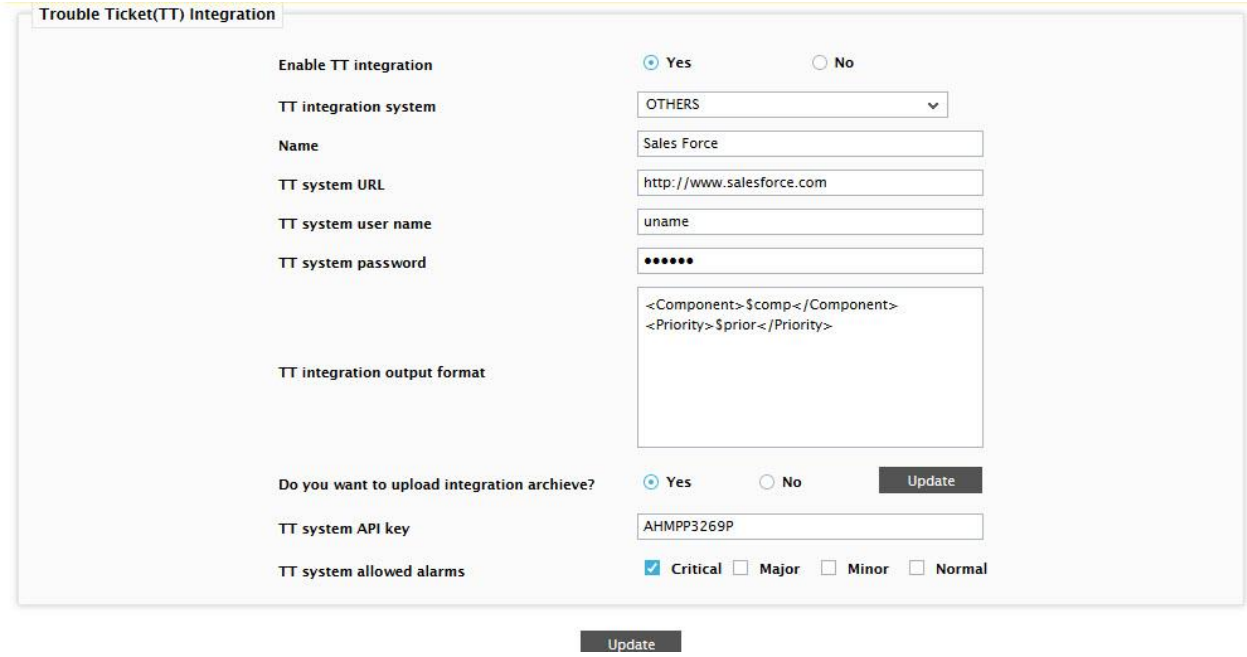

Figure 4.3: Integrating the eG manager with anu third-party TT system that supports a web services API

5. To enable integration via a web services interface, set the **Enable TT integration** flag in Figure 4.3 to **Yes**. By default, this is set to **No**.
6. From the **TT integration system** drop-down, select **OTHERS** as the TT system with which the eG manager should integrate.
7. Next, specify the **Name** of the TT system you want to integrate with.
8. Then, against the **TT system URL**, specify the Web Services Description Language (WSDL) URL via which the eG manager should connect to the web services interface of the third-party TT system. This URL should be of the following format that is recognized by the third-party system.
9. If the connection needs to be authenticated, then provide a valid user name and password against **TT system user name** and **TT system password** text boxes, respectively.
10. Then, against **TT integration output format**, describe the format in which alarm output is to be sent to the specified **TT system URL**.
11. To implement the web services-based integration of eG with a TT system, a Java class file needs to be written

and then jarred along with the following:

- Web services API files required for connecting to the TT system;
- A set of library files that support the Java class file containing the integration logic.

To know how to write this Java class file, contact support@eginnovations.com.

Once the Java class file is ready, make sure you jar it with the files mentioned above. eG refers to this jar file as the **TT integration archive file**. For the integration to work, the **TT integration ar** should be in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. To do so, first set the **Do you want to upload the integration archive?** flag to **Yes**, and click the **Update** button alongside. This will invoke Figure 4.4.

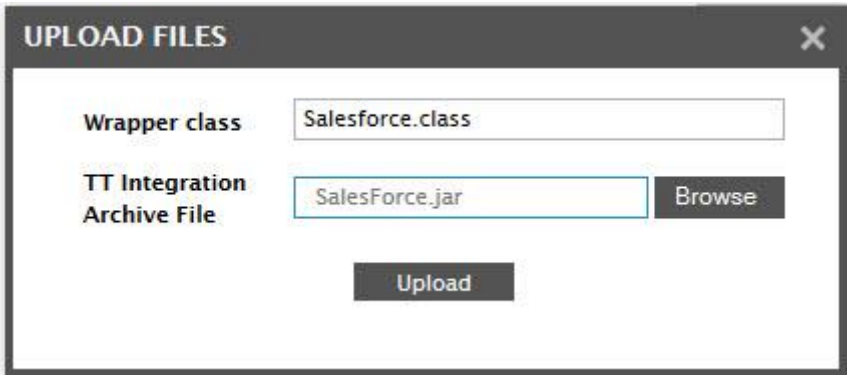

Figure 4.4: Uploading the TT integration jar file to the eG manager

Here, specify the name of the **Wrapper class** of the TT integration archive file, and provide the full path to the **TT Integration Archive File** to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button.

12. Then, specify the **TT system API key**. The authentication between the third-party TT system and the eG manager is through this API key. The eG manager will not be able to use the web service API of the TT system if this key is not specified or an invalid key is specified.
13. Then, indicate what type of eG alarms need to be forwarded to the TT system by selecting the check boxes corresponding to the alarm priorities listed in the **TT system allowed alarms**. Trouble tickets will be generated by the TT system only for the alarms of the chosen priorities.
14. Finally, click the **Update** button.

5

Conclusion

The eG Enterprise Suite has been specially designed keeping in mind the unique requirements of IT infrastructure operators. For more information on the eG family of products, please visit our web site at www.eginnovations.com.

For more details regarding eG Enterprise suite of products and the details of the metrics collected by the eG agents, please refer to the following documents:

- *Administering the eG Enterprise Suite*
- *Monitoring eG Enterprise*
- *The eG Installation Guide*
- *The eG Measurements Manuals*

We recognize that the success of any product depends on its ability to address real customer needs, and are eager to hear from you regarding requests for enhancements to the products, suggestions for modifications to the product, and feedback regarding what works and what does not. Please provide all your inputs as well as any bug reports via email to sales@eginnovations.com.